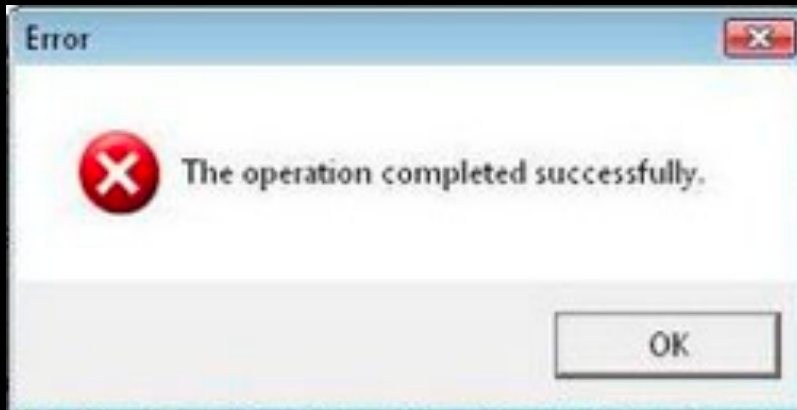




Graphics

Errors



What kind of bugs did you find in your code?



Reusing Variables



```
16 public void run() {  
17     int counter = 0;  
18     for (int i = 0; i < 10; i++) {  
19         int counter = counter + 1;  
20     }  
21 }  
22
```



Duplicate
variable age

You only need to tell Java the type of a variable once.



```
16 public void run() {  
17     int counter = 0;  
18     for (int i = 0; i < 10; i++) {  
19         counter = counter + 1;  
20     }  
21 }  
22  
23
```

Updating Variable Values

“equals”

=

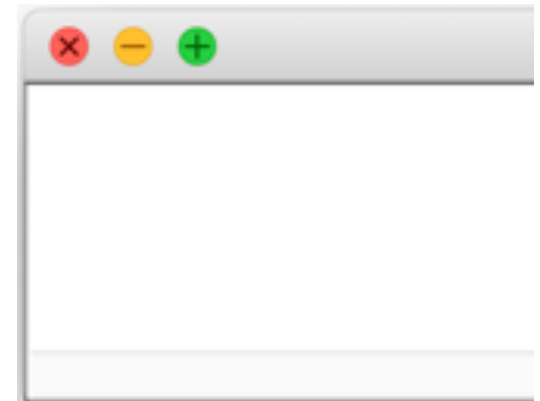


(1) Evaluate right hand side

(2) Store result in variable on left hand side



```
int year = 2019;  
int prevYear = year;  
year = year + 1;  
println(prevYear + " " + year);  
println(prevYear + year);
```



year

2019

prevYear

2019



Updating Variable Values

“equals”

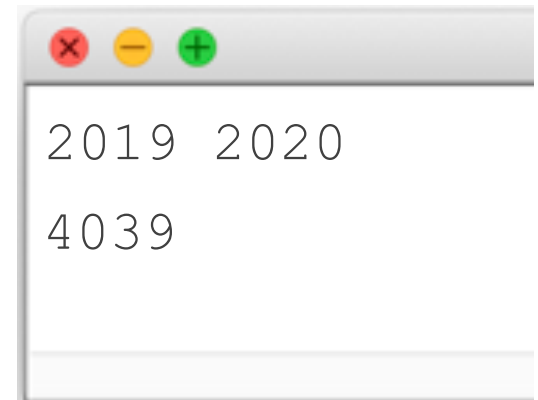
=



(1) Evaluate right hand side

(2) Store result in variable on left hand side

```
int year = 2019;  
int prevYear = year;  
year = year + 1;  
println(prevYear + " " + year);  
println(prevYear + year);
```

A terminal window with a grey title bar containing red, yellow, and green window control buttons. The terminal displays the output of the Java code: "2019 2020" on the first line and "4039" on the second line.

2019 2020
4039

year

2020

prevYear

2019



Updating Variable Values

“equals”

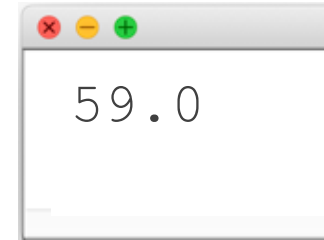
=



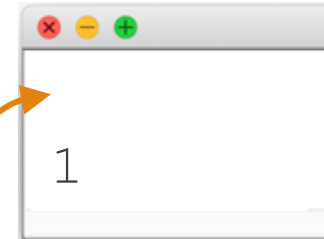
(1) Evaluate right hand side

(2) Store result in variable on left hand side

```
double seconds = 60;  
seconds--; seconds = seconds - 1;  
println(seconds);
```



```
int total = 15;  
total /= 10; total = total / 10;  
println();  
println(total);
```



```
int age;  
age = age + 1;
```



The local variable age may not have been initialized



We Called Karel, Maybe



I'm back! Maybe



Carly Rae Jepsen - Call Me Maybe - YouTube

Looping Beepers

How many beepers
did Karel put down?



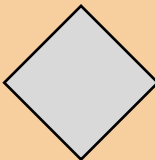
```
for (int i = 0; i < 5; i++) {  
    for (int j = 0; j < 4; j++) {  
        for (int k = 0; k < 3; k++) {  
            putBeeper();  
        }  
    }  
}
```

```
for (int i = 0; i < 3; i++) {  
    int countdown = 5;  
    while (countdown > 0) {  
        countdown--;  
        putBeeper();  
    }  
}
```


Looping Beepers

How many beepers
did Karel put down?



```
for (int i = 0; i < 5; i++) {  
    for (int j = 0; j < 4; j++) {  
          
    }  
}
```

3 x

60

```
for (int i = 0; i < 3; i++) {  
    int countdown = 5;  
    while (countdown > 0) {  
        countdown--;  
        putBeeper();  
    }  
}
```

Looping Beepers

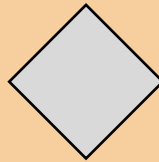
How many beepers
did Karel put down?



```
for (int i = 0; i < 5; i++) {
```

4 x

3 x



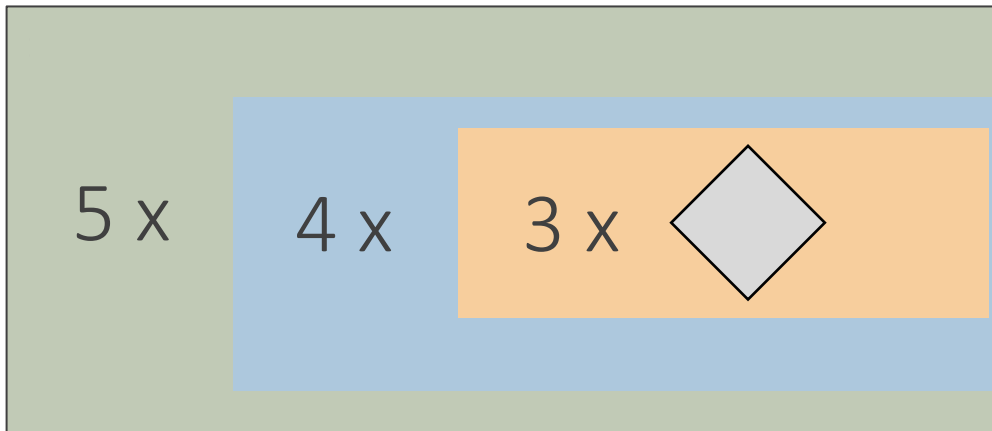
```
}
```

60

```
for (int i = 0; i < 3; i++) {  
    int countdown = 5;  
    while (countdown > 0) {  
        countdown--;  
        putBeeper();  
    }  
}
```

Looping Beeper

How many beepers
did Karel put down?



60

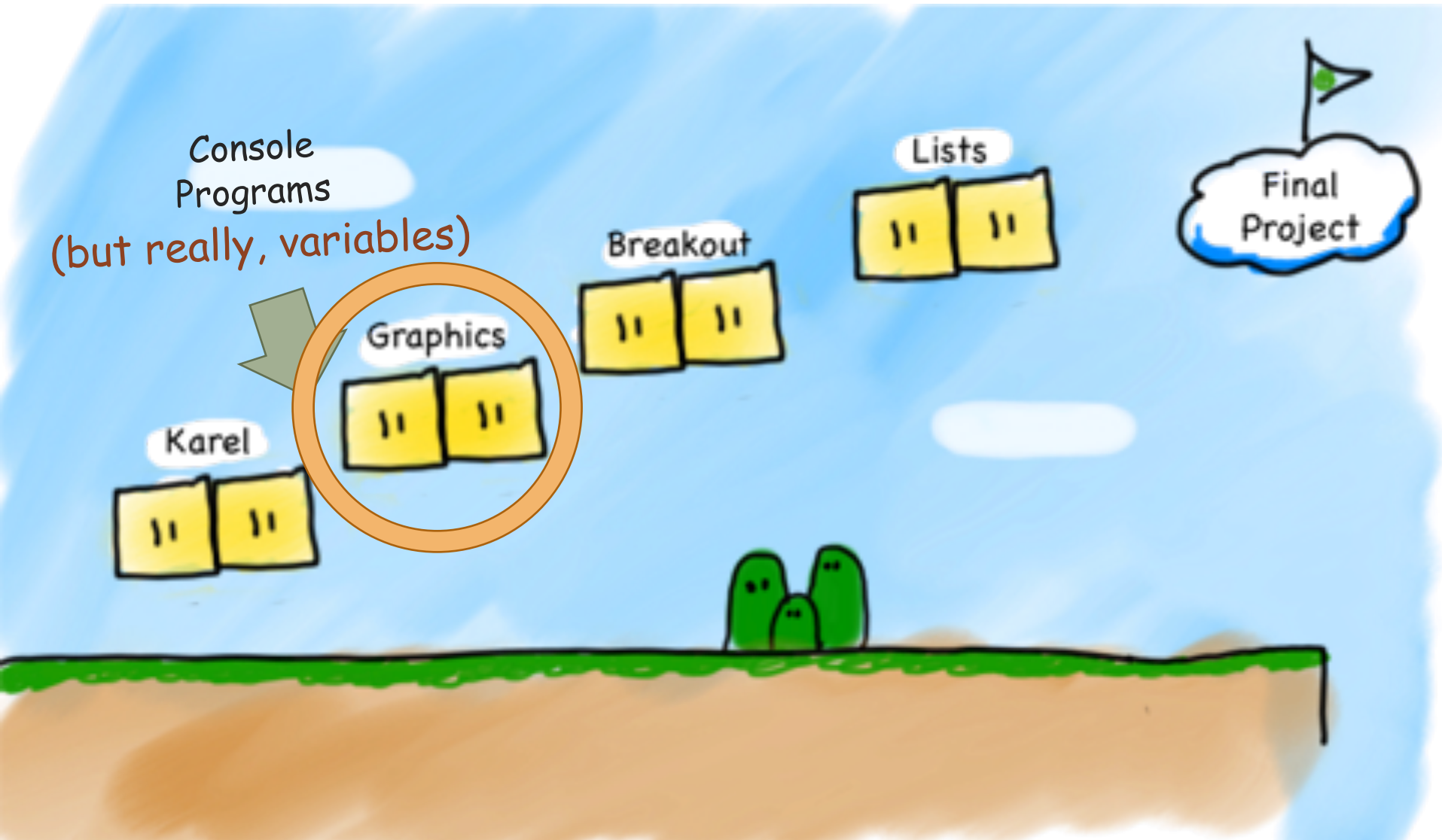
```
for (int i = 0; i < 3; i++) {  
    int countdown = 10;  
    while (countdown > 0) {  
        countdown -= 2;  
        putBeeper();  
    }  
}
```

15

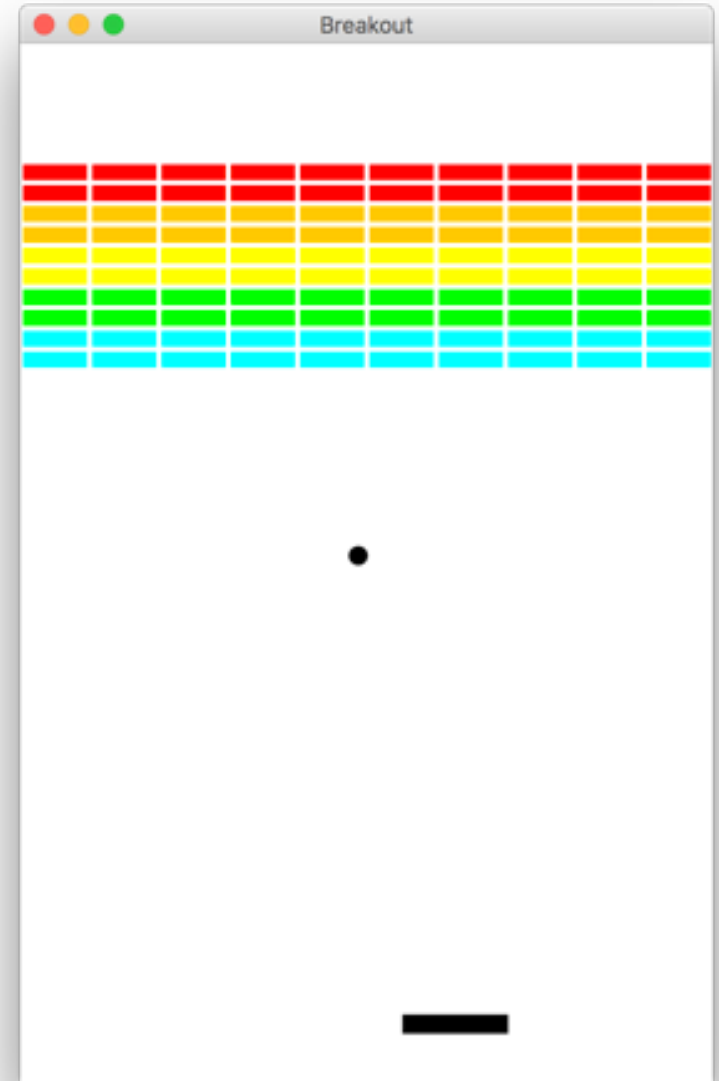
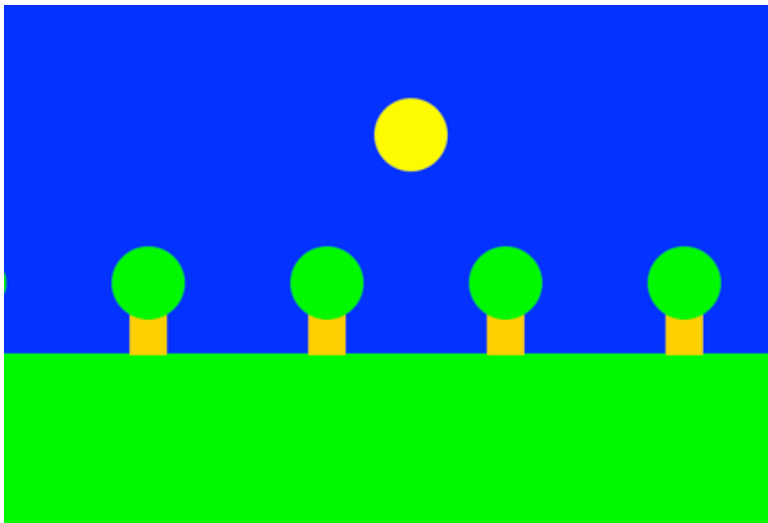
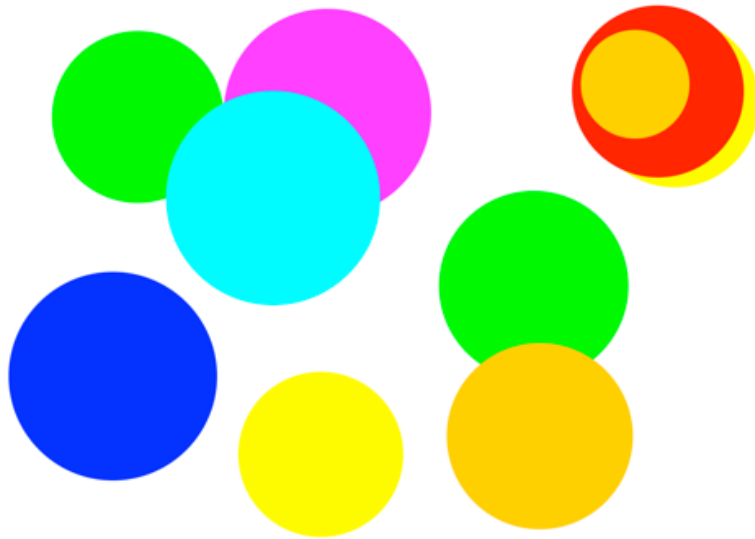
Programming takes practice.



Our Second Step



Beyond Console Programs

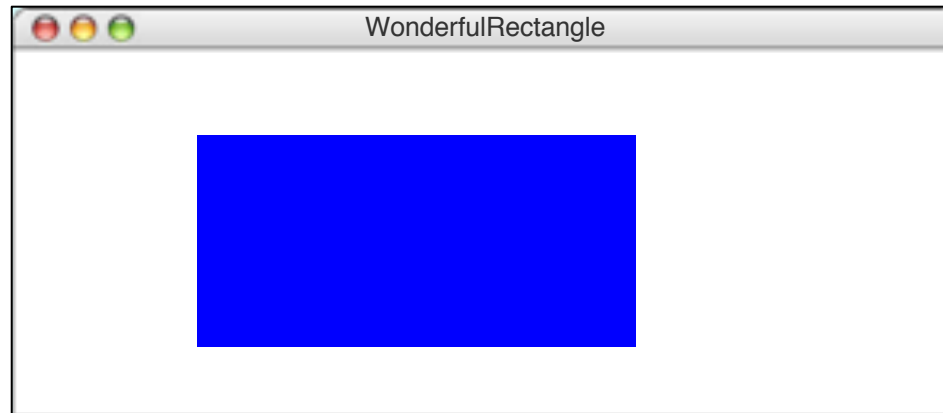


GRect

GRect is a variable that stores a rectangle.

As an example, the following `run` method displays a rectangle

```
public class WonderfulRect extends GraphicsProgram {  
    public void run() {  
        GRect rect = new GRect(220, 120);  
        rect.setFilled(true);  
        rect.setColor(Color.BLUE);  
        add(rect, 50, 50);  
    }  
}
```

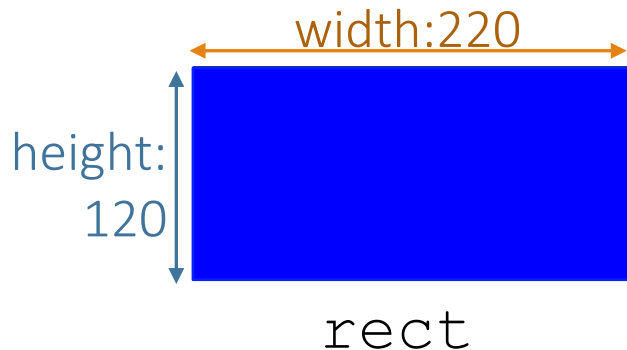


GRect

GRect is a variable that stores a rectangle.

As an example, the following `run` method displays a rectangle

```
public class WonderfulRect extends GraphicsProgram {  
    public void run() {  
        GRect rect = new GRect(220, 120);  
        rect.setFilled(true);  
        rect.setColor(Color.BLUE);  
        add(rect, 50, 50);  
    }  
}
```



GRect

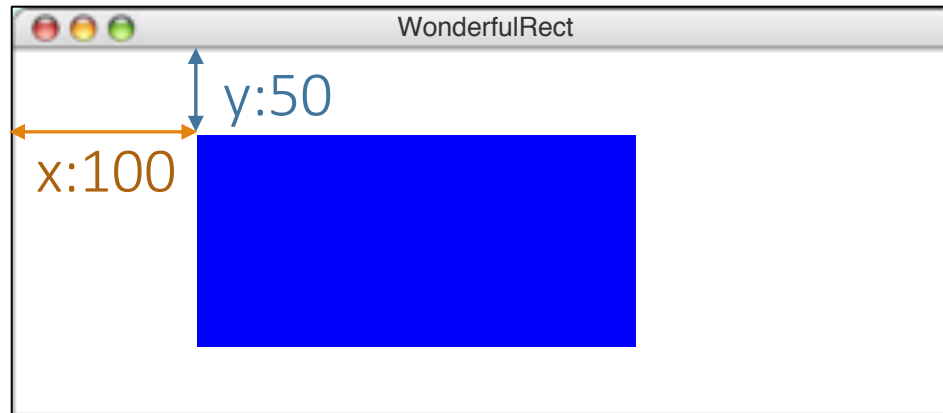
GRect is a variable that stores a rectangle.

As an example, the following `run` method displays a rectangle

```
public class WonderfulRect extends GraphicsProgram {  
    public void run() {  
        GRect rect = new GRect(200, 200);  
        rect.setFilled(true);  
        rect.setColor(Color.BLUE);  
        add(rect, 100, 50);  
    }  
}
```

! You must call
`add()` to display things!

Coordinates
for a rectangle
are the **top left**
corner.



Graphics Coordinates

0,0

x 40,20

x 120,40

x 40,120

getWidth();

getHeight();

Two GRects

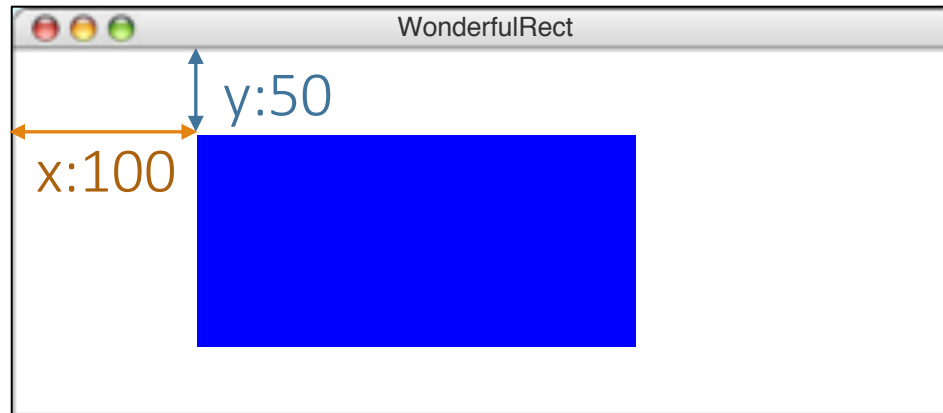
GRect is a variable that stores a rectangle.

As an example, the following `run` method displays a rectangle

```
public class WonderfulRect extends GraphicsProgram {  
    public void run() {  
        GRect rect = new GRect(200, 200);  
        rect.setFilled(true);  
        rect.setColor(Color.BLUE);  
        add(rect, 100, 50);  
    }  
}
```

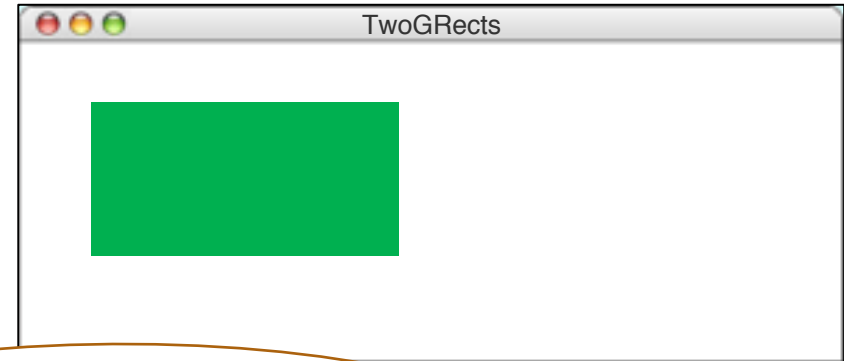
! You must call
`add()` to display things!

Coordinates
for a rectangle
are the **top left**
corner.



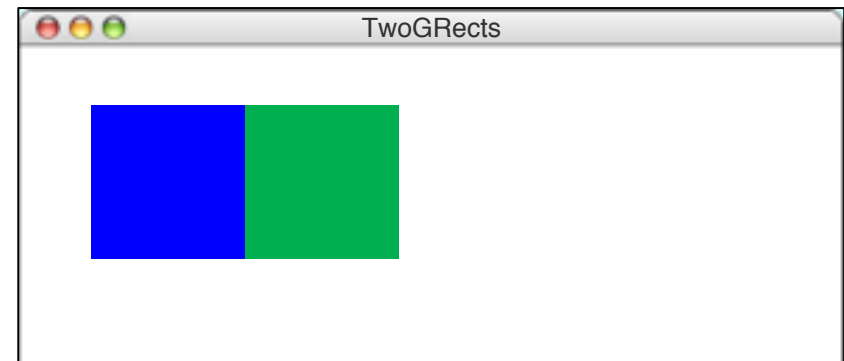
Two GRects

```
GRect rect = new GRect(100, 100);  
rect.setFilled(true);  
rect.setColor(Color.BLUE);  
add(rect, 50, 50);  
rect.setColor(Color.GREEN);  
add(rect, 150, 50);
```



! multiple add() commands just move things around!

```
GRect rectB = new GRect(100, 100);  
rectB.setFilled(true);  
rectB.setColor(Color.BLUE);  
add(rectB, 50, 50);  
  
GRect rectG = new GRect(100, 100);  
rectG.setFilled(true);  
rectG.setColor(Color.GREEN);  
add(rectG, 150, 50);
```

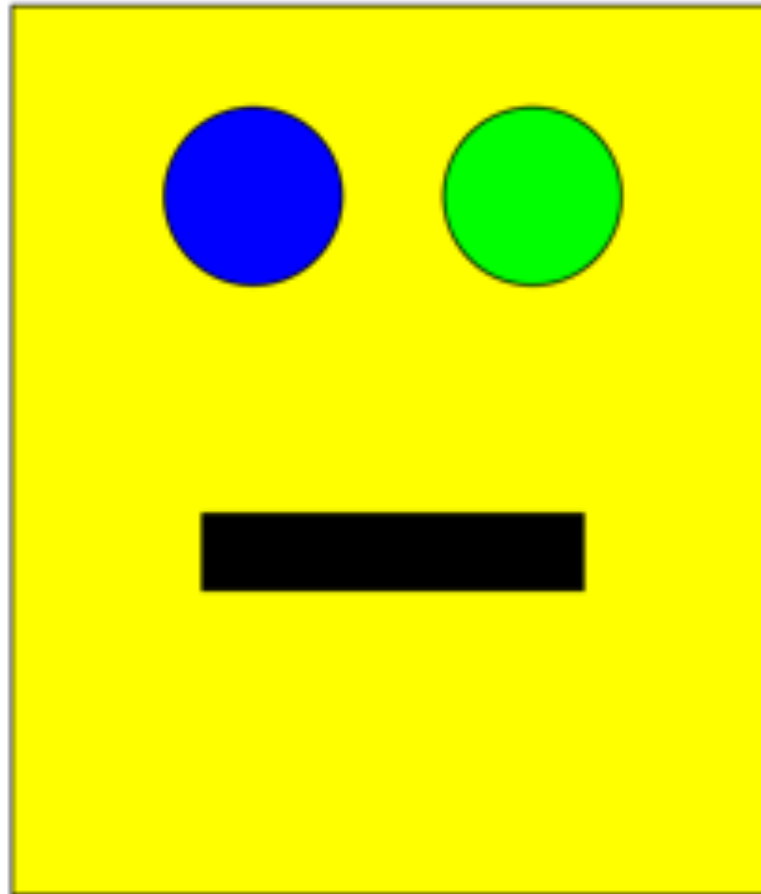


This draws two separate rectangles.

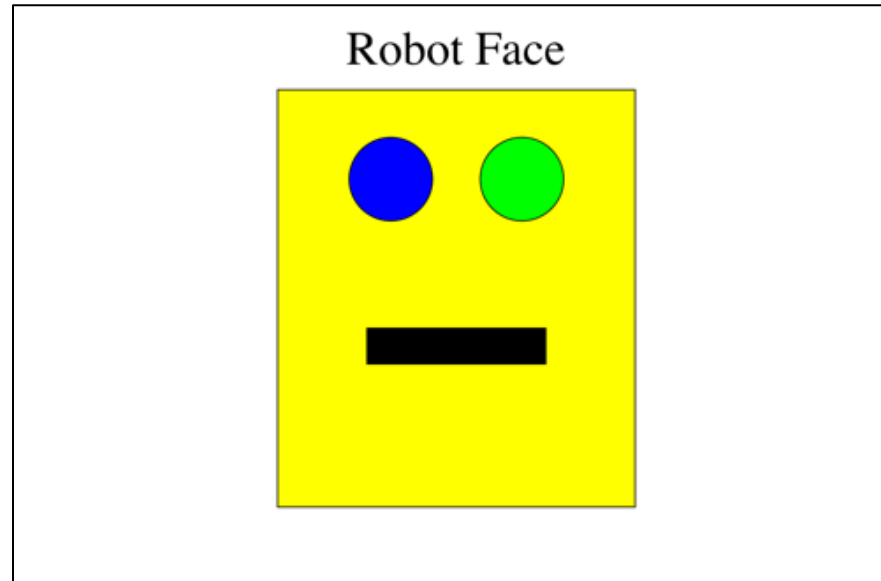
Questions?

Karel's Grandpa

Robot Face



Karel's Grandpa



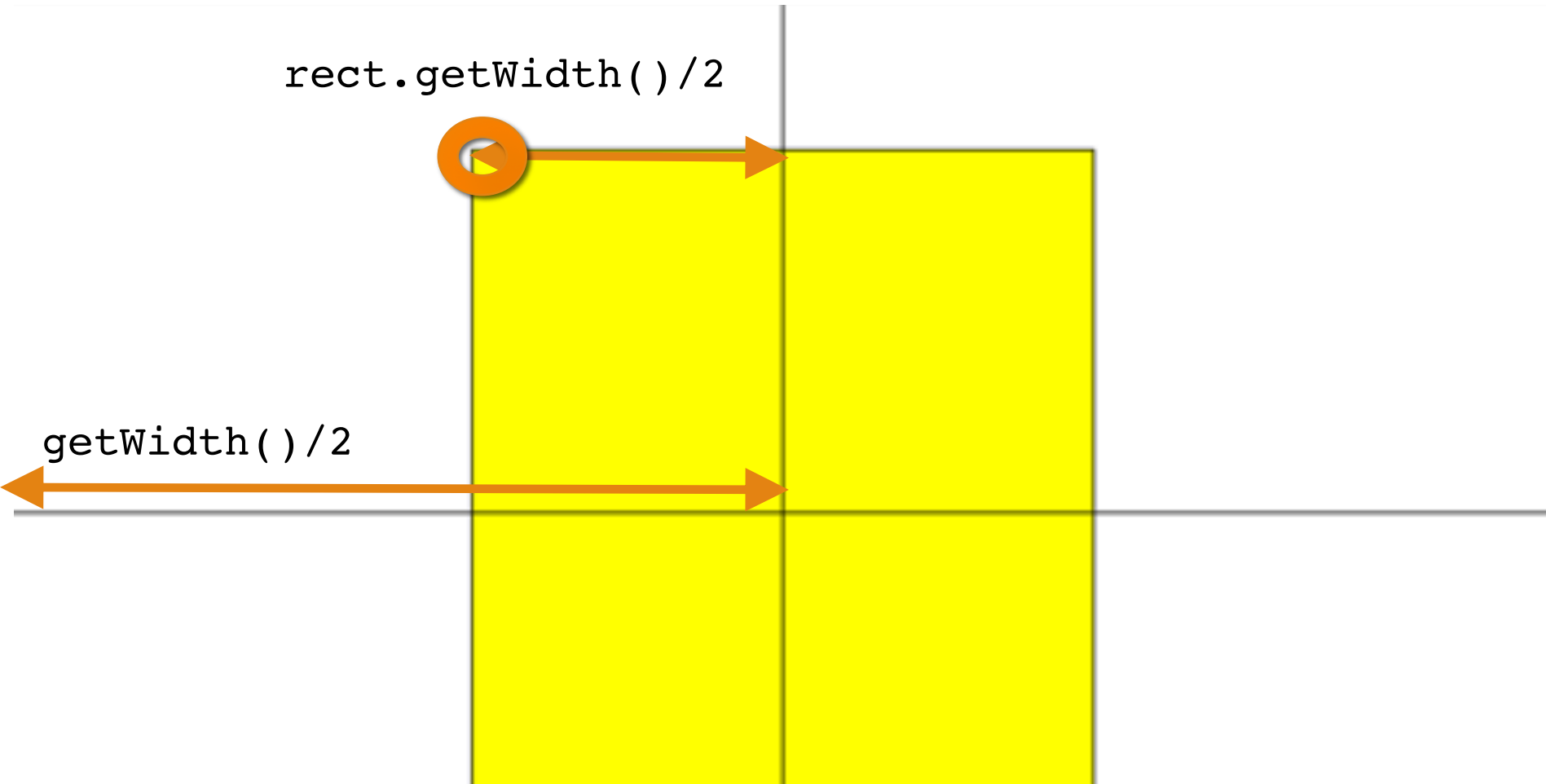
```
public void run() {  
    drawHead();  
    drawMouth();  
    drawLabel();  
    drawEyes();  
}
```

drawHead()

$(x, y)?$



drawHead()

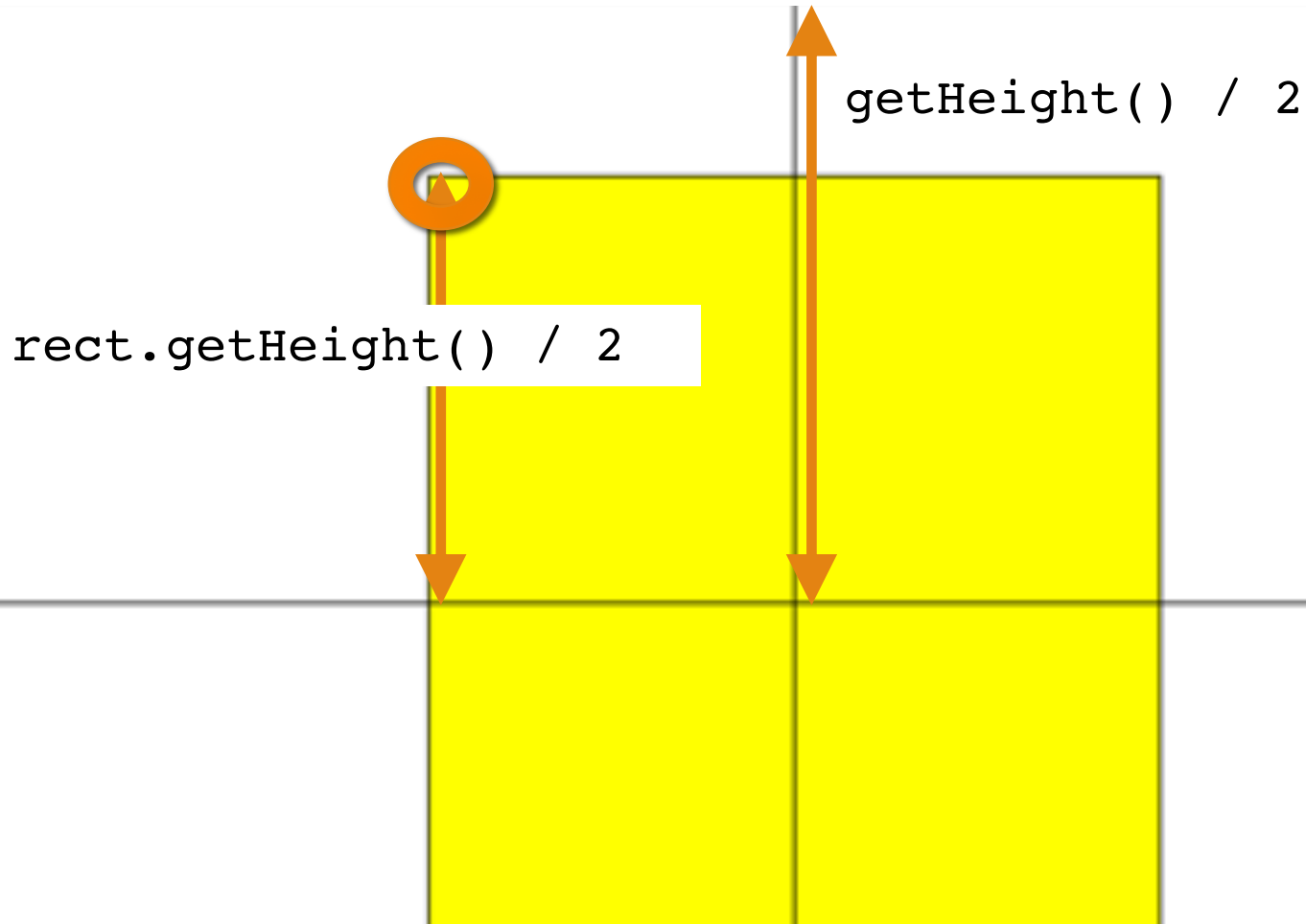


```
double headX = getWidth() / 2 - rect.getWidth() / 2;
```

or

```
double headX = (getWidth() - rect.getWidth()) / 2;
```

drawHead()



```
double headY = getHeight() / 2 - rect.getHeight() / 2;
```

or

```
double headY = (getHeight() - rect.getHeight()) / 2;
```

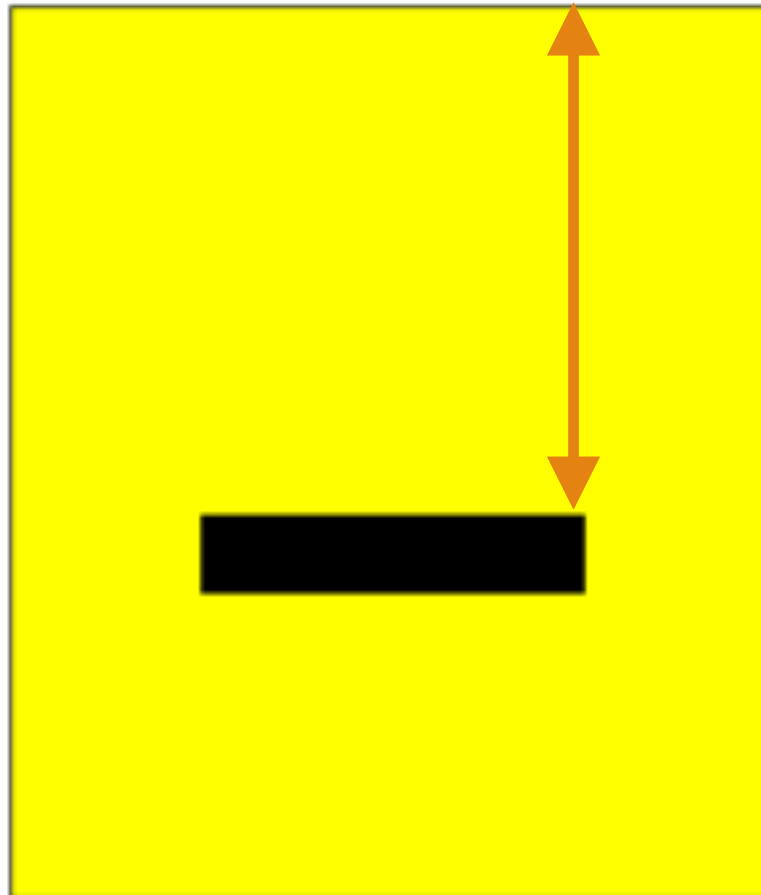
Constants

Once constants are defined, they cannot be changed anywhere.
It has scary syntax, but is easy to use.

```
public class RobotFace extends GraphicsProgram {  
  
    /* The distance from the top of the head  
       * to the top of the mouth */  
    private static final int MOUTH_Y_OFFSET = 200;  
  
    public void run() {  
        drawHead();  
        drawMouth();  
        drawLabel();  
        drawEyes();  
    }  
}
```

drawMouth()

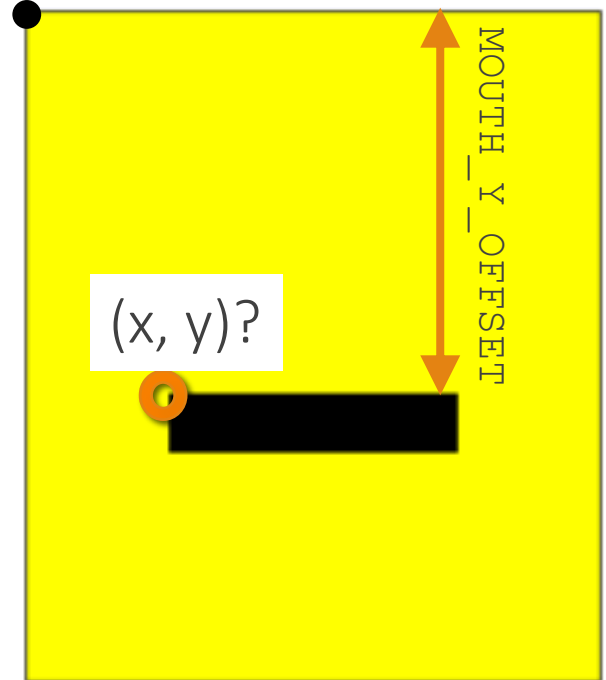
```
/* The distance from the top of the head to the top of the mouth */  
private static final int MOUTH_Y_OFFSET = 200;
```



drawMouth()

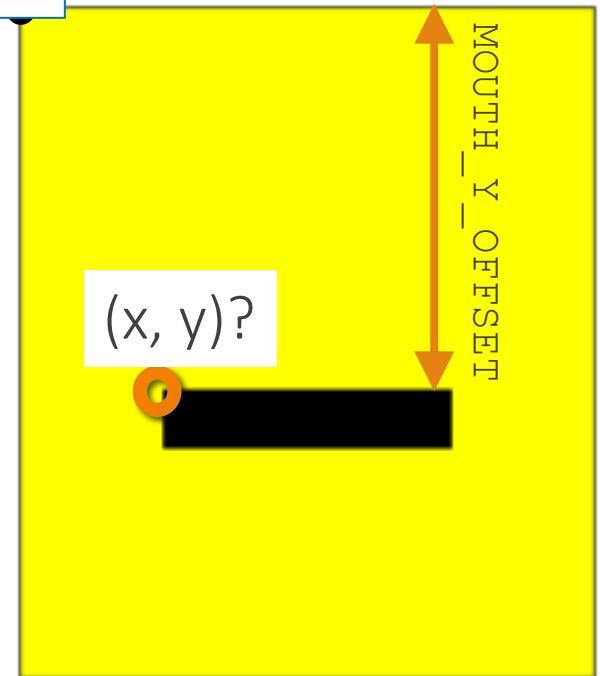
```
public class RobotFace extends GraphicsProgram {  
  
    private static final int MOUTH_Y_OFFSET = 200;  
    ... // more constants, hidden  
  
    private void drawMouth() {  
        double headX = ...;  
        double headY = ...;  
  
        double mouthX = ...  
        double mouthY = ????????  
  
        GRect mouth = new GRect(  
            mouthX, mouthY,  
            MOUTH_WIDTH, MOUTH_HEIGHT);  
        ??????????  
        ??????????  
    }  
    ... // more code  
}
```

(headX, headY)

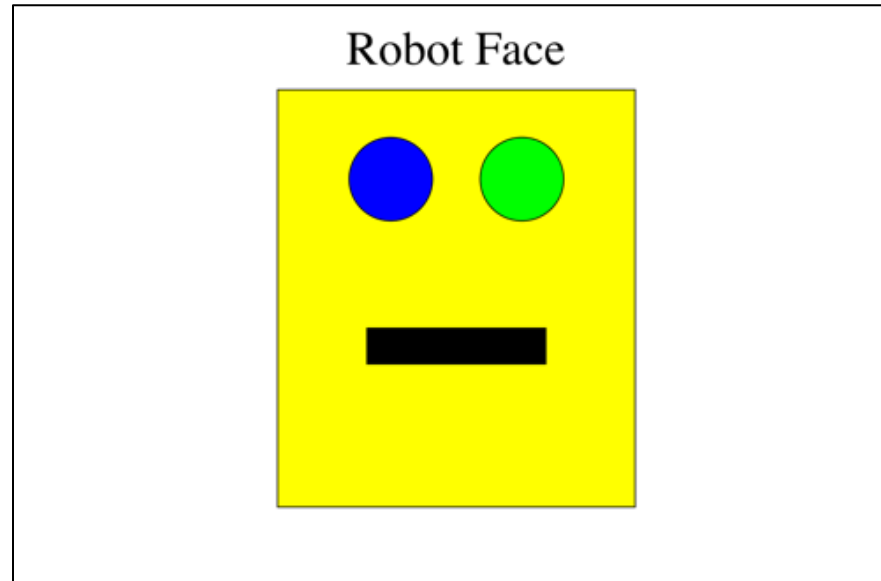


drawMouth()

```
public class RobotFace extends GraphicsProgram {  
  
    private static final int MOUTH_Y_OFFSET = 200;  
    ... // more constants, hidden  
  
    private void drawMouth() {  
        double headX = ...;  
        double headY = ...;  
  
        double mouthX = ...;  
        double mouthY = headY + MOUTH_Y_OFFSET;  
  
        GRect mouth = new GRect(  
            mouthX, mouthY,  
            MOUTH_WIDTH, MOUTH_HEIGHT);  
        mouth.setFilled(true);  
        add(mouth);  
    }  
    ... // more code  
}
```



Karel's Grandpa



```
public void run() {  
✓    drawHead();  
✓    drawMouth();  
    drawLabel();  
    drawEyes();  
}
```

GLabel

A variable that represents text.

```
public class HelloProgram extends GraphicsProgram {  
    public void run() {  
        GLabel label = new GLabel(  
            "hello, world", 100, 75);  
        label.setFont("SansSerif-36");  
        label.setColor(Color.RED);  
        add(label);  
    }  
}
```



GLabel

A variable that represents text.

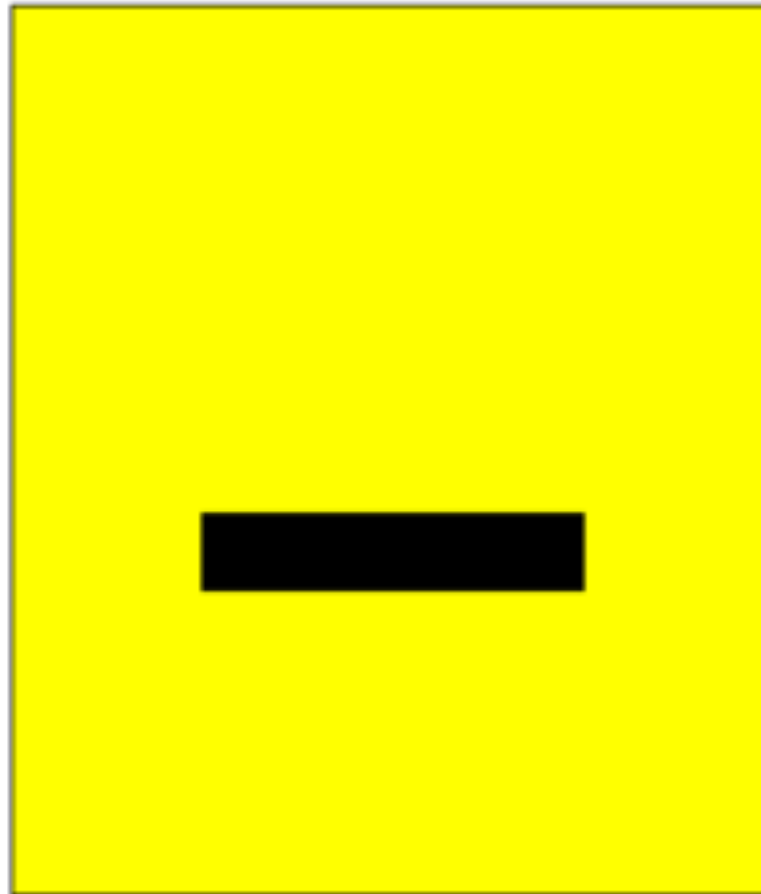
```
public class HelloProgram extends GraphicsProgram {  
    public void run() {  
        GLabel label = new GLabel(  
            "hello, world", 100, 75);  
        label.setFont("SansSerif-36");  
        label.setColor(Color.RED);  
        add(label);  
    }  
}
```



drawLabel()

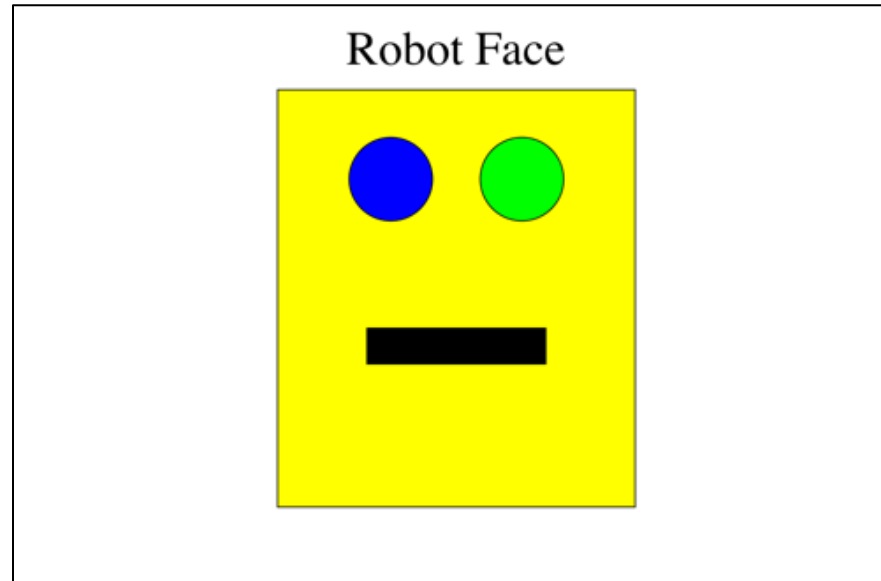


Robot Face



```
/* The distance from the top of the screen to the base of the label */  
private static final int LABEL_Y = 50;
```

Karel's Grandpa



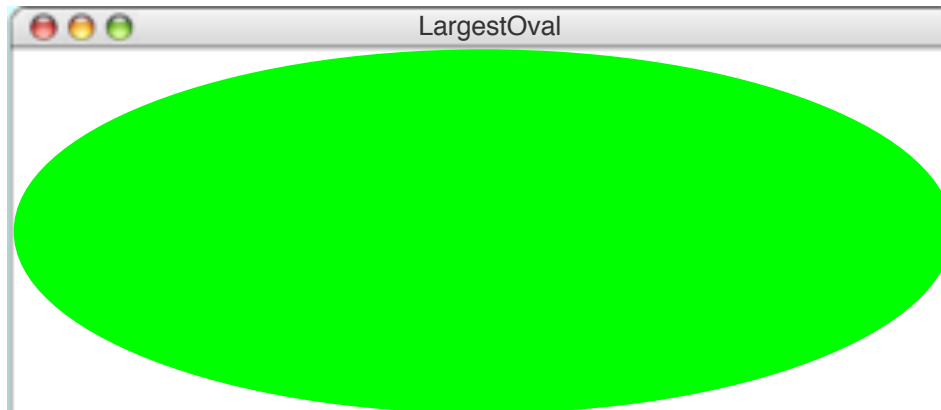
```
public void run() {  
✓    drawHead();  
✓    drawMouth();  
✓    drawLabel();  
    drawEyes();  
}
```

GOval

The `GOval` class represents an elliptical shape defined by the boundaries of its enclosing rectangle.

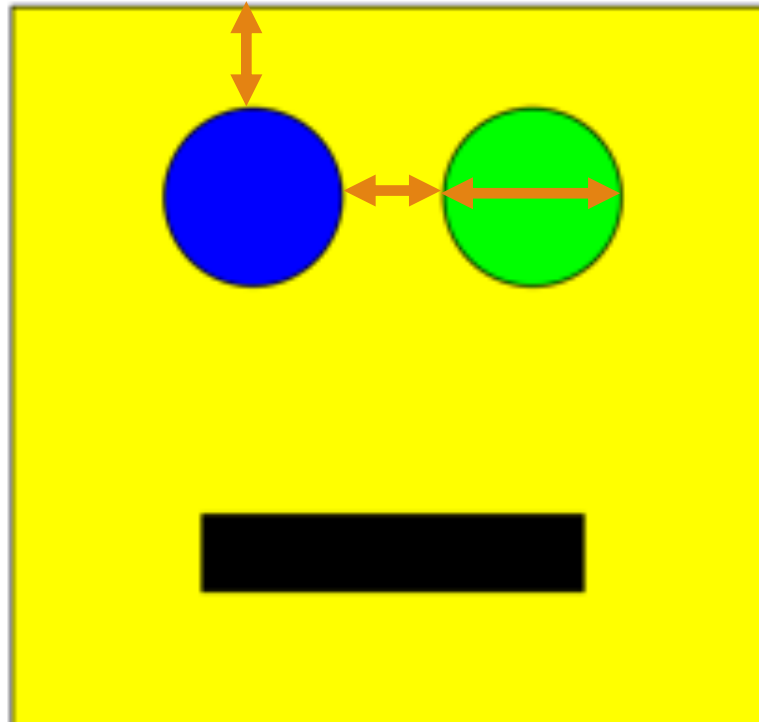
As an example, the following `run` method creates the largest oval that fits within the canvas:

```
public void run() {  
    GOval oval = new GOval(getWidth(), getHeight());  
    oval.setFilled(true);  
    oval.setColor(Color.GREEN);  
    add(oval, 0, 0);  
}
```



drawEyes()

Robot Face

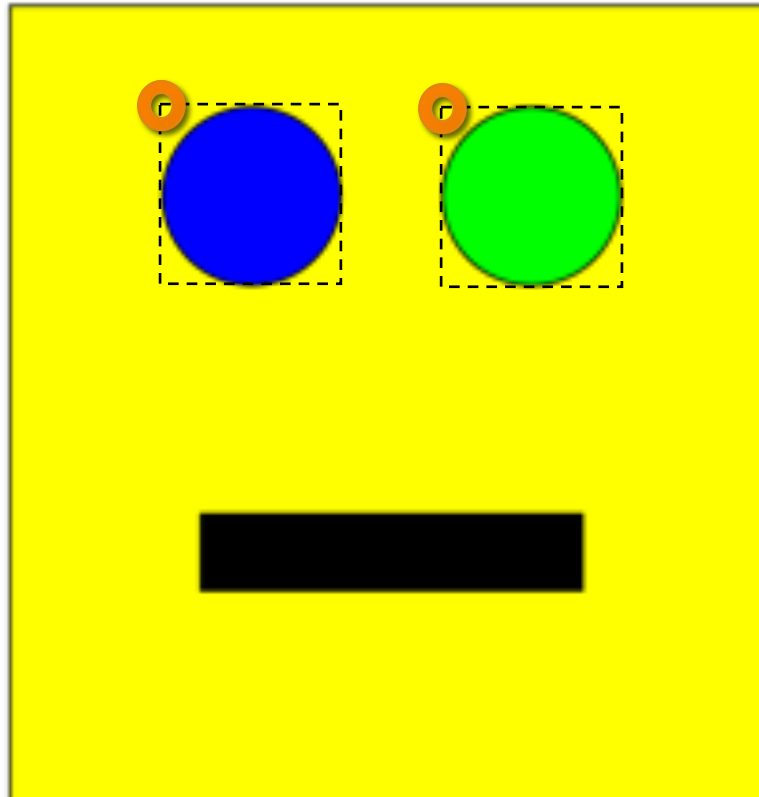


```
/* The diameter of each eye*/  
private static final int EYE_DIAMETER = 70;  
/* The distance from the top of the head to the top of the eyes*/  
private static final int EYE_Y_OFFSET = 40;  
/* The distance in between the two eyes */  
private static final int EYE_X_SEPARATION = 40;
```

drawEyes()

Robot Face

(leftX, eyeY)?
(rightX, eyeY)?

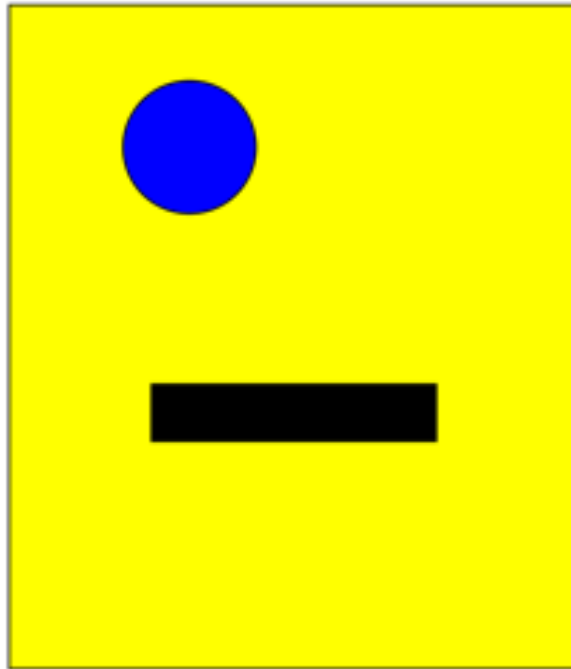


```
double eyeY = headY + EYE_Y_OFFSET;  
double leftX = getWidth()/2 - EYE_X_SEPARATION/2 - EYE_DIAMETER;  
double rightX = getWidth()/2 + EYE_X_SEPARATION/2;
```

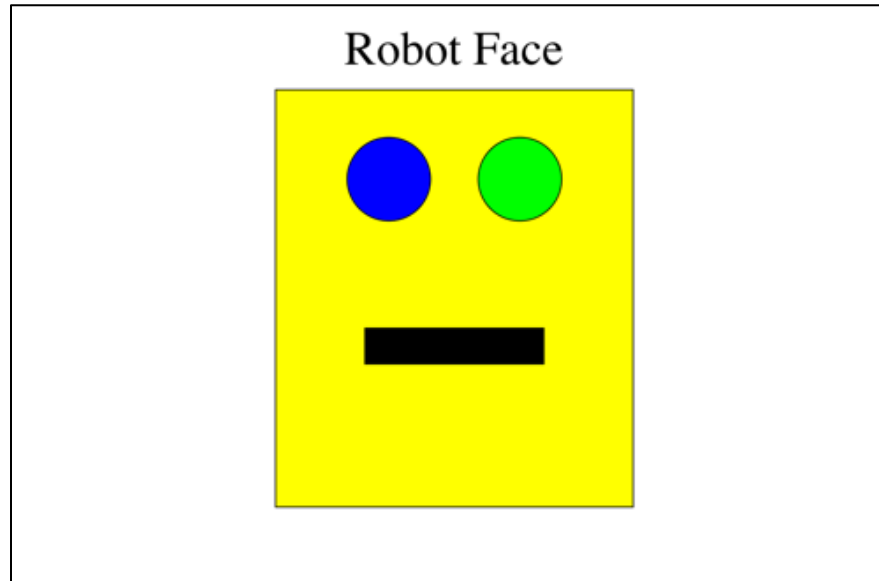
drawEyes()

Programming time!

Robot Face

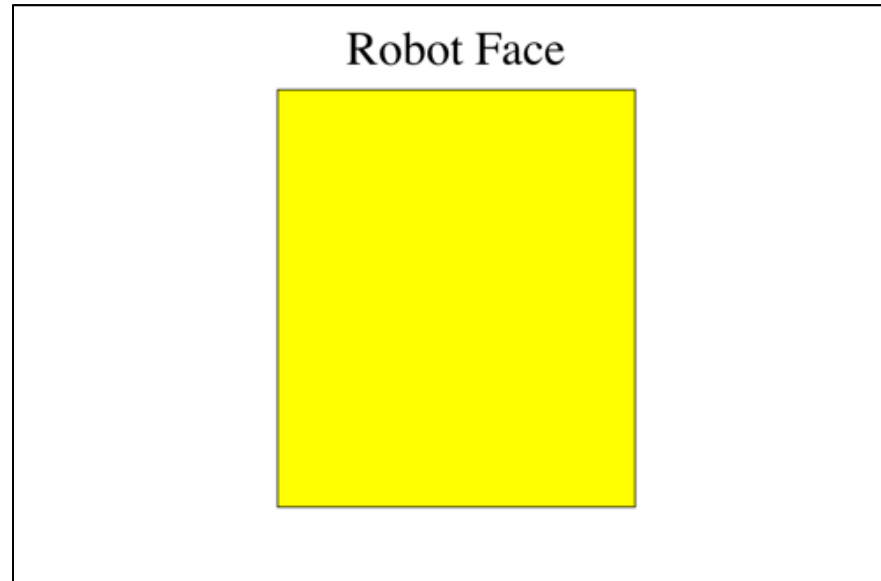


Karel's Grandpa



```
public void run() {  
    ✓ drawHead();  
    ✓ drawMouth();  
    ✓ drawLabel();  
    ✓ drawEyes();  
}
```



Not Karel's Grandpa



```
public void run() {  
    drawMouth();  
    drawLabel();  
    drawEyes();  
    drawHead();  
}
```

! Order of
add() matters!


Karel's Grandpa

CS Bridge Handouts ▾ Projects ▾ Examples ▾ Slides ▾ Bonus ▾ Forms ▾  

Draw Robot Face

Problem written by Eric Roberts. Updated by UN Karel Yan.

Write a program that draws this picture. You can use the constants provided, but feel free to change them (or recolor any part of the face).

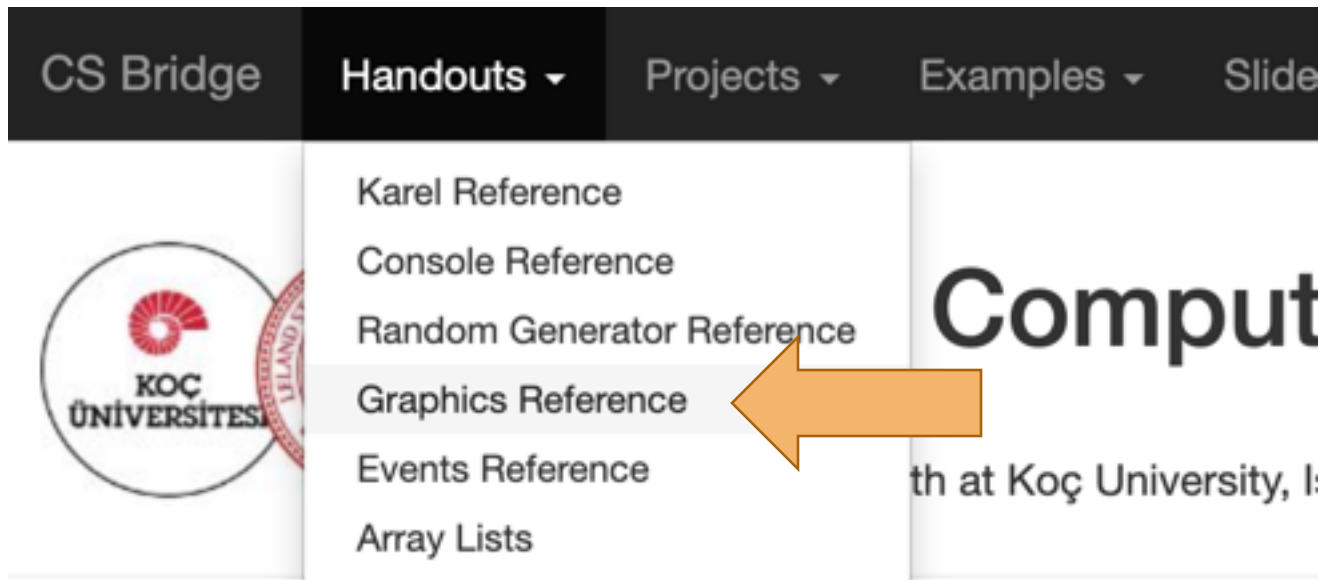


Solution

```
/**
 * Robot Face
 * -----
 * Draws an awesome robot face, with label and different colored
 * eyes!!!@#$%&!
 */
public class RobotFace extends GraphicsProgram {
```

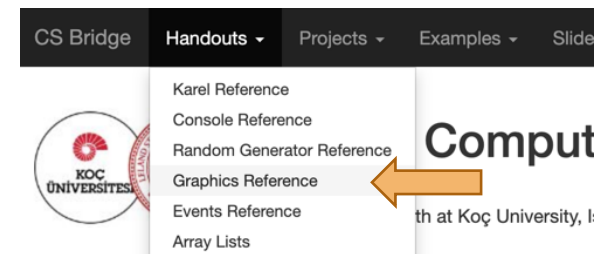
Read the solution code if you get stuck!

Graphics Methods Reference



Graphics Methods

<code>add(object)</code>	Adds the object to the canvas at the front of the stack
<code>add(object, x, y)</code>	Moves the object to (x, y) and then adds it to the canvas
<code>remove(object)</code>	Removes the object from the canvas
<code>removeAll()</code>	Removes all objects from the canvas
<code>getElementAt(x, y)</code>	Returns the frontmost object at (x, y) , or <code>null</code> if none
<code>getWidth()</code>	Returns the width in pixels of the entire canvas
<code>getHeight()</code>	Returns the height in pixels of the entire canvas
<code>setBackground(c)</code>	Sets the background color of the canvas to c .
<code>pause(milliseconds)</code>	Pauses the program for the specified time in milliseconds
<code>waitForClick()</code>	Suspends the program until the user clicks the mouse



Reference Sheet

Constructors

new GLabel(String text) or new GLabel(String text, double x, double y)

Creates a new **GLabel** object; the second form sets its location as well.

new GRect(double x, double y, double width, double height)

Creates a new **GRect** object; the **x** and **y** parameters can be omitted and default to 0.

new GOval(double x, double y, double width, double height)

Creates a new **GOval** object; the **x** and **y** parameters can be omitted and default to 0.

new GLine(double x1, double y1, double x2, double y2)

Creates a new **GLine** object connecting (**x1, y1**) and (**x2, y2**).

Methods common to all graphical objects

void setLocation(double x, double y)

Sets the location of this object to the specified coordinates.

void move(double dx, double dy)

Moves the object using the displacements **dx** and **dy**.

double getWidth()

Returns the width of the object.

double getHeight()

Returns the height of the object.

void setColor(Color c)

Sets the color of the object.

Methods available for GRect and GOval only

void setFilled(boolean fill)

Sets whether this object is filled (**true** means filled, **false** means outlined).

boolean isFilled()

Returns **true** if the object is filled.

void setFillColor(Color c)

Sets the color used to fill this object. If the color is **null**, filling uses the color of the object.

Methods available for GLabel only

void setFont(String fontName)

Sets the font, as described in Chapter 5.

double getAscent()

Returns the height above the text baseline.

double getDescent()

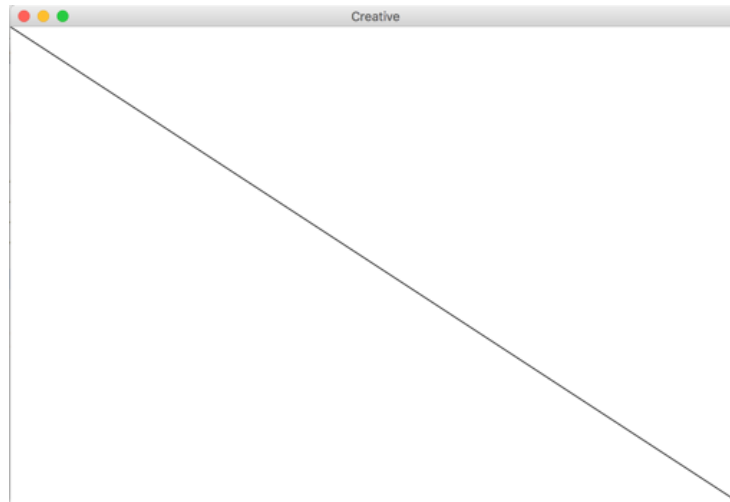
Returns the height below the text baseline.

GLine

The `GLine` class represents a line defined by a start point and an end point.

As an example, the following `run` method creates a diagonal line across the canvas:

```
public void run() {  
    GLine line = new GLine(0,0, getWidth(), getHeight());  
    add(line);  
}
```



(bonus) Karel's Grandpa

